

REMARKS

Claims 1-20 are currently pending, of which claims 1, 2, 7, 8, 12, 16, and 20 are amended herein. In the Office Action mailed February 17, 2004, the Examiner objected to claim 2 based on an informality. The Examiner rejected claims 1 and 3-6 under 35 U.S.C. 102(a) as anticipated by Bonola (U.S. Pat. No. 5,978,858). The Examiner rejected claims 16-20 under 35 U.S.C. 102(a) as anticipated by Aronson (U.S. Pat. No. 6,128,673). The Examiner rejected claims 2, 7, and 8-15 under 35 U.S.C. 103(a) as unpatentable over Bonola in view of Wooten (U.S. Pat. No. 5,832,492). Applicant traverses these rejections for the following reasons.

1. Response to Objection to Claim 2

The Examiner objected to claim 2 based on the informality that the acronym "TD" was not defined upon first use. Claim 2 has been amended herein to define the acronym upon first use. Applicant therefore respectfully requests that this objection be withdrawn.

2. Bonola Does Not Anticipate Independent Claim 1**a. Introduction**

The Examiner rejected independent claim 1 under 35 U.S.C. 102(a) as anticipated by Bonola. Claim 1 is directed to a "method for transferring data over a Universal Serial Bus (USB)." As currently amended and with emphasis added, the method comprises the steps of "*polling a burst communication adapter device coupled to the USB using a first type of channel for the burst communication adapter device; receiving a reply message from the burst communication adapter device, the reply message indicating that the burst communication adapter device has data for transfer via a second type of channel for the burst communication*

adapter device; and *responsive to receiving the reply message, issuing a bulk channel read request* for the burst communication adapter device.”

Bonola is directed toward a “packet protocol and distributed burst engine,” with which a host system processor may transmit commands and/or data to an I/O device, as well as receive data from the device, whether requested or not. These commands and data take the form of linked lists of packets in an area of a host system memory 112; this area of memory 112 is accessible by both the host system processor 100 and a distributed burst engine (DBE) 206. (Bonola, Fig. 2A, col. 4:63–5:57.) In fact, for each I/O device (“bus master”) 118, Bonola teaches forming four linked lists of packets in host system memory 112, two of which are referred to as a solicited packet pool 200 and an unsolicited packet pool 202. (Id.) Both the processor 100 and the DBE 206 add packets to and remove packets from these pools, to facilitate the exchange of commands and data between the host system processor 100 and the I/O device 118. (Id.)

Applicant respectfully submits that Bonola neither teaches nor suggests (i) polling a burst communication adapter device; nor (ii) issuing a bulk channel read request for the device *responsive to receiving a reply message from the device, the reply message indicating that the device has data to transfer via a second type of channel.* As Bonola fails to teach or suggest all elements of Applicant’s independent claim 1, it is therefore allowable.

**b. Bonola Neither Teaches Nor Suggests
Polling a Burst Communication Adapter Device**

Applicant respectfully submits that Bonola neither teaches nor suggests “polling a burst communication adapter device.” The Examiner’s citation to Bonola, col. 9:10-20, pertains to the host system processor 100 polling a subset of these packets *in the host system’s own memory 112.* (Bonola, Figs. 1, 2A, 4; col. 4:15-18, 4:63–5:4, 8:30–9:5.) This is in contrast to the method

of claim 1, which comprises "polling a burst communication adapter device." In fact, the teaching of Bonola is that the sharing of an area of host system memory 112 between the host system processor 100 and the distributed burst engine (DBE) 206 "eliminates the need to poll the I/O device 118 directly and keeps the processor 100 off of the host bus 102". (Bonola, col. 17:16-18.) Thus, Bonola teaches away from polling a burst communication adapter device.

The Examiner's citation to Bonola, col. 9:10-20, with respect to the polling of packets teaches that "[p]ackets 250 may be submitted either asynchronously (A), polled (P) or interrupt (I)." This does not refer to polling a burst communication adapter device, but to what Bonola calls a "submission type." (Bonola, Fig. 3, col. 6:44-64, 7:32-59.) When the host system processor 100 adds a packet to the solicited packet pool 200 for an I/O device 118, the processor designates the packet as having one of three possible submission types (asynchronous, polled, or interrupt) by setting certain bits in a "Va" field 270 of the packet's header. (Id.) It is this "polled" submission type to which the Examiner's citation of col. 9:10-20 refers.

A packet's submission type determines whether the processor 100 will be notified once the I/O device 118 has completed the request associated with the packet and, if so, in what manner the processor will be notified. (Bonola, col. 8:30-57.) After being processed by the DBE 206, packets are "completed" in different ways, depending on their submission type. (Id.) If asynchronous, the processor 100 receives no notification of completion of the packet, such as for a graphics command. (Id.) If interrupt, the processor receives "completion notification by means of a hardware interrupt asserted by the I/O device 118." (Bonola, col. 8:36-38.)

In the case of polled packets, as cited by the Examiner, the DBE 206 returns the packet to memory 112 after submitting the request to the I/O device 118. (Bonola, col. 8:36-38, 17:4-18.)

Thereafter, the processor 100 will periodically *poll the returned packet in memory 112* until the

DBE 206 has updated the packet to reflect completion of the request. (Id.) Thus, Bonola does not teach polling a burst communication adapter device; on the contrary, Bonola teaches away, stating that "direct communication between the processor 100 and PCI bus master 118 is reduced or limited," and teaching a "decoupling of the processor 100 and PCI bus masters 118" resulting in "eliminated processor 100 reads from target I/O devices 208." (Bonola, 5:49-57.)

c. Bonola Neither Teaches Nor Suggests Issuing a Bulk Channel Read Request for the Burst Communication Adapter Device Responsive to Receiving a Reply Message

Applicant respectfully submits that Bonola neither teaches nor suggests "responsive to receiving the reply message, issuing a bulk channel read request for the burst communication adapter device," where the reply message indicates "that the burst communication adapter device has data for transfer via a second type of channel for the burst communication adapter device." For this element, the Examiner cites the "solicited request" shown in Bonola, Figure 4. Bonola does not, however, teach or suggest issuing such a request (or any request) in response to any particular event or message; rather, Bonola teaches only that solicited requests are initiated by host software 220 running on host system processor 100. (Bonola, col. 8:59-9:46) Applicant therefore respectfully submits that Bonola does not anticipate claim 1.

In Bonola, data is received from an I/O device in one of two ways, corresponding to the two ways Bonola teaches for categorizing the packets described above: (1) solicited requests, as cited by the examiner, and (2) unsolicited requests. (Bonola, col. 4:64-5:12.) Of the two, only a solicited request would ever even be in response to some event as unsolicited requests are, by definition, not requested. In Bonola, the solicited requests are made by host software but are not made in response to receiving a reply message - let alone a reply message from a burst communication adapter device as is claimed by Applicant.

In operation, both the processor 100 and the DBE 206 access the packets in host system memory 112. With respect to solicited requests, Bonola teaches that, "[a]s requests are solicited from host software 220, the device driver 230 unlinks packets 250 from the head H of the free queue 272 [(the 'solicited packet pool 200')], fills in the packet's payload 254 and links the packets 250 to the tail T of the request queue 274." (Bonola, col. 8:59-9:46) The DBE 206 then accesses the packets, and "parses [the packets] to determine what operation was requested by the [device] driver 230." (Bonola, col. 8:18-29.) Note that Bonola refers interchangeably to processor 100 and device driver 230. (Bonola, col. 6:36-43.) No mention is made in Bonola, however, of such a request being issued responsive to any particular message or event.

Unsolicited requests arise when data arrives spontaneously at an I/O device 118, such as mouse movements or data that arrives at a network adapter. (Bonola, col. 5:7-13.) In such cases, the host system processor 100 is made aware of the unsolicited data when the DBE 206 adds a packet to the unsolicited packet pool 202, after which the host system processor 100 accesses the packets and processes the associated data. (Bonola, col. 9:40-46.) This does not implicate a bulk channel read request, as the unsolicited requests are, by definition, not requested.

Thus, unsolicited requests are not relevant, and, as to solicited requests, Bonola teaches only that these requests are initiated by host software 220; Bonola does not teach or suggest, however, that solicited requests are initiated in response to any particular message or event, much less to receiving a reply message from an I/O device.

Thus, Bonola neither teaches nor suggests claim 1. Claim 1 is therefore allowable. Accordingly, dependent claims 2-7 are also allowable.

3. Aronson Does Not Anticipate Claims 16-20

The Examiner rejected claim 16 under 35 U.S.C. 102(a) as anticipated by Aronson. Applicant respectfully submits that Aronson does not anticipate claim 16; among other reasons, Aronson neither teaches nor suggests: "wherein upon receipt of an ethernet packet addressed to said universal serial bus communications adapter, said universal serial bus communications adapter transmits a data present signal via said interrupt channel process."

With respect to this element, the Examiner cites (i) the "packet received bit" shown in Aronson, Fig. 4, after the first state 110 as the "data present signal" and (ii) element 116 of the same Fig. 4 with respect to transmitting the "data present signal via said interrupt channel process." The example of digital-protocol translation disclosed throughout the specification of Aronson is translation between the USB digital protocol and the Ethernet digital protocol, and vice versa. Fig. 4, cited by the Examiner, pertains to a process 108 implemented by a state machine 100. (See Aronson, col. 6:33-34.) State machine 100 is part of a USB controller 94, which is part of an SIE (Serial Interface Engine) Interface 76. (See Aronson, Figs. 2, 3, 3A; col. 4:50-55; col. 6:10-7:28.) SIE Interface 76 is part of a first protocol circuitry 52. (Id.) In Aronson, the example given of the first protocol is USB. (Id.)

Thus, the process 108 of Aronson, Fig. 4 pertains to processing USB packets, not ethernet packets as in claim 16. Specifically, the process 108 is directed to either (i) receiving a USB packet via a USB port 66 from a USB device 68, and transmitting that USB packet to a memory 60 using Direct Memory Access (DMA); or (ii) accessing a USB packet from memory 60 using DMA, and transmitting that USB packet to USB device 68 via USB port 66. (Id.) Process 108 does not pertain to the receipt of an ethernet packet, and thus the applicant respectfully submits that Aronson does not anticipate claim 16.

Furthermore, although both Aronson and claim 16 discuss interrupts, the nature of these interrupts is very different, illustrating that Aronson does not anticipate claim 16. In Aronson, once an ethernet packet is received via ethernet port 70, an interrupt is set for the microprocessor 58, which is integrated into Aronson's digital-protocol translator. (Aronson, Figs. 2, 7, 7B; col. 4:16-25; 4:61-67; 8:4-11; 9:1-35). When the processor 58 processes this interrupt, the processor 58 then sends the data received via the ethernet packet out to a USB device 68 in the form of USB data via USB port 66. (Aronson, Fig. 8; col. 36-43; 10:4-8 ("USB end point 2 interrupt").)

Thus, the interrupt in Aronson is a notification to the microprocessor 58, which is internal to the digital-protocol translator; this notification then facilitates the transmission of data that came in through Ethernet port 70 out through USB port 66. In Aronson, then, it is the data that is programmatically transmitted out the USB port 66, rather than a data present signal via an interrupt channel process of a USB driver, as is the case with claim 16.

In claim 16, once an ethernet packet is received into the USB communications adapter, the adapter transmits a data present signal via the interrupt channel process of the USB driver. This interrupt is not internal to the USB communications adapter, as is the case in Aronson; rather, it is the data present signal itself that is transmitted using the USB physical layer to a USB host device with which the USB physical layer is adapted for communication. And the data present signal is sent via the USB interrupt channel from the adapter to the host device.

In summary, Aronson discloses receiving an ethernet packet, and setting an interrupt for the processor internal to the digital-protocol translator, to facilitate passing the received ethernet data out the USB port. In contrast, claim 16 teaches receiving an ethernet packet, and sending a data present signal out the USB port (using the USB interrupt channel); this USB interrupt will then be processed on a host device in the normal course of processing USB interrupt packets

(rather than by a microprocessor internal to the adapter or digital-protocol translator). The host device will thereby be notified that the adapter of claim 16 has data to transfer to the host device.

Thus, Aronson neither teaches nor suggests claim 16. Claim 16 is therefore allowable. Accordingly, dependent claims 17-20 are also allowable.

4. **Bonola and Wooten Do Not Anticipate Claims 8-15**

The Examiner rejected claim 8 under 35 U.S.C. 103(a) as being unpatentable over Bonola in view of Wooten. Applicant respectfully submits that claim 8 is allowable over Bonola in view of Wooten for reasons essentially identical to those given above with respect to claim 1. For one, Applicant has responded above to the Examiner's citation of Bonola, col. 9:10-20, with respect to the polling of packets. Briefly stated, Bonola discloses polling packets in memory that have already been processed to an I/O device, where the polling is to determine whether the task associated with the packet has been completed. Claim 8, however, includes a polling message sent by a class driver executing on a microprocessor to a burst communication adapter device (an I/O device). The polling message is sent via a first type of channel of the USB via a host controller. As shown above, Bonola teaches away from this type of polling.

Furthermore, claim 8 includes a "class driver being...configured to receive the reply message and, responsive thereto, create a transfer descriptor and attach the transfer descriptor to the predetermined endpoint descriptor list." However, Wooten teaches that inputting data from an I/O device occurs by processing a "token packet" to the device, specifying the direction of data transfer, as well as "the type of transfer, the serial bus device address and the endpoint number." (Wooten, col. 6:37-40, 44-45.) "The source of the transfer then sends a data packet or indicates it has no data to transfer." (Wooten, col. 6:45-47.) Thus, the host system in Wooten processes the token packet to the I/O device whether or not the I/O device has data to transfer.

Therefore, Wooten does not teach creating a transfer descriptor and attaching the transfer descriptor to the predetermined endpoint descriptor list in response to receiving a reply message from the I/O device, where that reply message indicates that the device has data to transfer via a second type of channel.

Thus, claim 8 is allowable. Accordingly, dependent claims 9-11 are also allowable. Independent claim 12 includes similar elements and is therefore also allowable along with dependent claims 13-14.

5. Conclusion

Prompt notice of allowance is respectfully requested. Should the Examiner have any questions, the Examiner is encouraged to contact the Applicant's attorney, Brian Harris, at his direct dial number of 312-913-3303.

Respectfully submitted,

Date: 6/17/04



Brian R. Harris
Reg. No. 45,900
Attorney for Applicant

McDonnell Boenhen Hulbert & Berghoff LLP
300 South Wacker Drive
Chicago, IL 60606
312-913-0001

McDonnell Boenhen Hulbert & Berghoff LLP
300 South Wacker Drive
Chicago, IL 60606
312-913-0001

-16-